TEMA 3: *"Mètodes i funcions"*

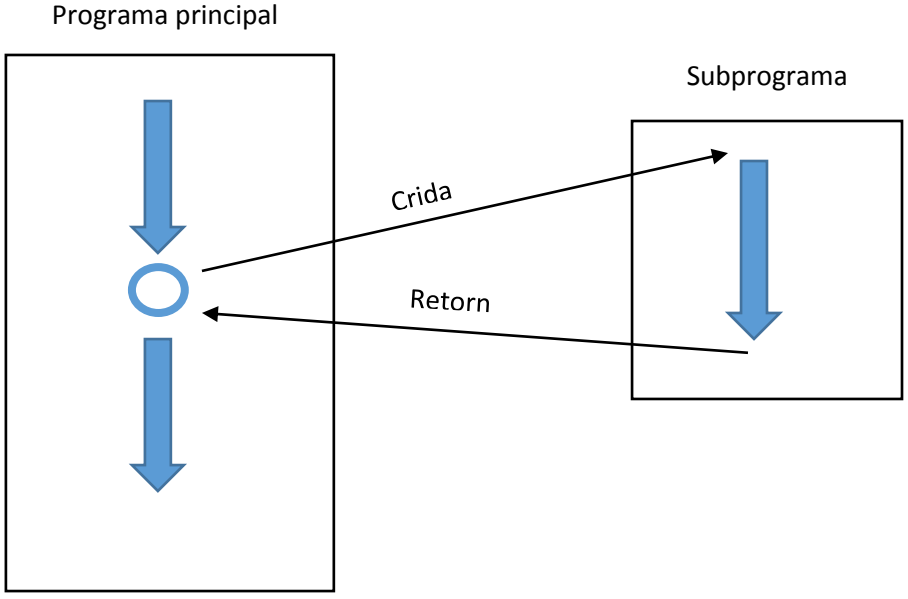
BASE TEÒRICA

Un determinat problema complex es pot dividir en problemes més senzills, els quals a la seva vegada poden ser de nou dividits fins arribar a la descripció de subproblemes molt simples. Aquesta filosofia o metodologia de treball s'anomena **Refinament Progressiu o por passes, disseny descendent, Programació Top-Down** o també "**Divideix i guanyaràs".**
En el mon de la programació, aquests "subproblemes" s'anomenen "mòduls", funcions, subprogrames, subrutines o procediments, i concretament en Java, mètodes.
El llenguatge Java és un llenguatge modular, es a dir, un programa es pot dividir en varis mòduls, cadascú dels quals realitza una tasca determinada, això dona lloc al que es coneix com *programació modular* i que es constitueix com el paradigma de programació que és l'evolució natural de la *programació estructurada*. Un mètode o funció és per tant un petit programa o mini programa que s'utilitza en un programa. És un conjunt d'instruccions que es poden invocar des de qualsevol part d'un programa i tantes vegades com sigui necessari.

Els mètodes malgrat afegeixen certa complexitat en la seva utilització, ens proporcionen diverses avantatges:

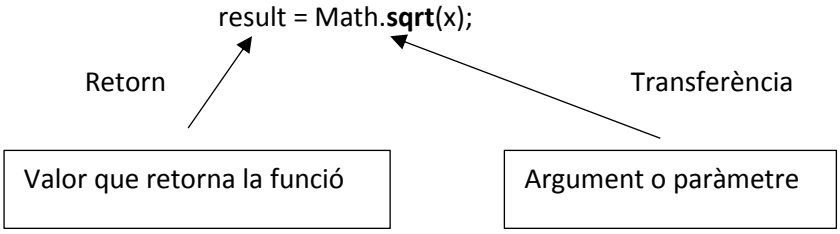
- Permeten implementar independentment tasques concretes (abstracció: caixa negra)
- Simplifiquen un programa, gràcies a poder descompondre'l en parts més petites (tècnica que es coneix com "*disseny top-down*".
- més fàcil d'escriure, llegir i segui un programa (llegibilitat)
- facilitat la portabilitat del codi (reutilització)

Un mètode té implícit un mecanisme de transferència i retorn del control del programa i les dades, es a dir, quan des d'un punt d'un programa es "crida" un mètode, des d'aquest es transfereix el control al programa cridat per que un cop fet aquest segon es retorni al primer:



- ***Estructura d'un mètode***

Fins ara, hem utilitzat mètodes (funcions) estàndard, es a dir definides en un paquet (biblioteca), com per exemple:



Es parla de funcions quan no van associades a un objecte en concret. Com a Java tot son classes parlarem sempre de mètodes, es a dir, funcions associades a objectes concrets. Per que un mètode funcioni com una funció, es a dir no estigui associat a cap classe ni objecte, haurà de tenir el modificador ***static*** que fa que no es requereixi d'un objecte per poder utilitzar-la.

Java ens permet definir els nostres propis mètodes. Un mètode "especial" i que sempre ha de ser, ja que és el més important i des de el que comença sempre qualsevol programa i controla tot el desenvolupament posterior, és el mètode **main**.

Un mètode es definirà segons la següent sintaxis:

```
<tipus_acces> <tipus_resultat>  <nom_de_la_funció>  ( llista_de_paràmetres )
{
        cos_de_la_funció ;
        return  <expressió> ;  //ha de ser del tipus  <tipus_resultat>
}
```

On: **<tipus_acces>** : modificadors per determinat qui i com pot accedir al mètode, poden ser una o dues paraules. Normalment serà *public static*, de manera que es podrà accedir al mètode des de qualsevol altra classe (qualsevol programa).

<tipus_resultat> : tipus de dada que retorna la funció com a resultat de la instrucció retorn, en el cas particular que no hi hagi *return* el tipus serà *void.* En Java pot ser qualsevol tipus de dada, fins i tot un objecte definit mitjançant una classe (POO).

<expressió> : valor que retorna la funció. Amb l'ordre *return* acaba la funció.

<llista de paràmetres> : variables formals d'entrada amb el seu tipus, també anomenades *arguments* del mètode. Aquestes variables copien els valors dels arguments utilitzats des de la crida de la funció. En el cas que no hi hagi paràmetres, es deixa el parèntesi en blanc o es posa la paraula clau *void*.

return : sempre que el tipus de resultat no sigui *void* hi haurà un *return* mitjançant el qual el mètode proporciona la seva sortida a l'entrada de dades proporcionada. Un *return* acaba el mètode i retorna el valor proporcionat per aquest com a sortida. Si el mètode té com a tipus de resultat *void*, no tindrà *return*.

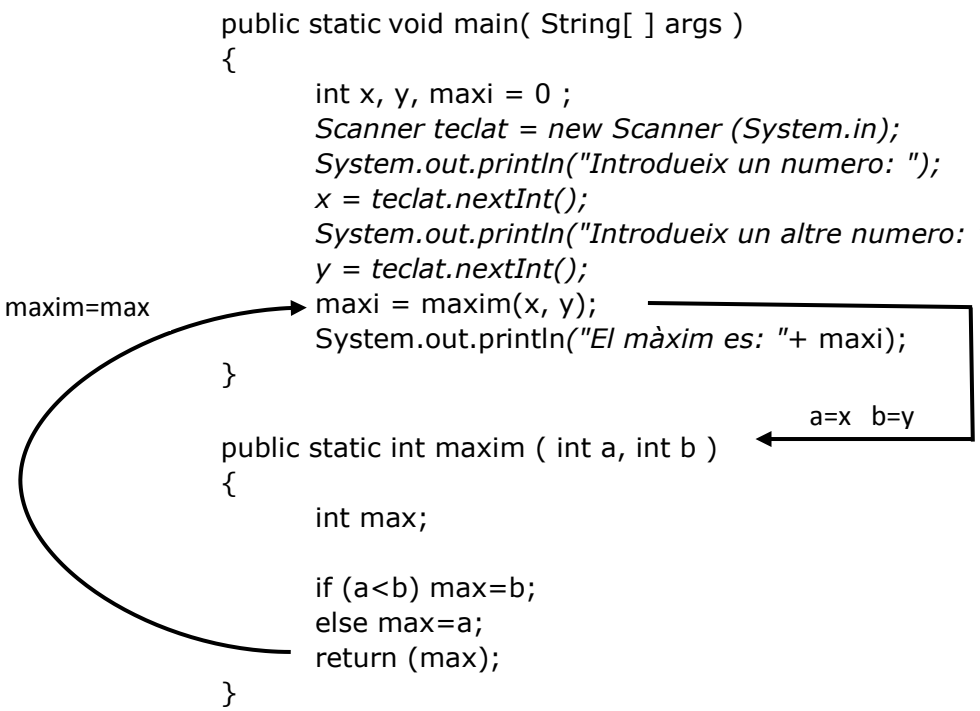
Per exemple, una funció serà:

```
public static int maxim (int a, int b)
{
        int max;
        if (a<b) max=b;
        else max=a;
        return max;
}
```

Un cop implementat un mètode, ja es pot utilitzar. Per poder utilitzar-lo s'haurà d'invocar el mètode, "cridar-lo". La "crida" provocarà l'execució de totes les instruccions que hi hagi en el cos del mètode amb la substitució efectiva dels seus paràmetres, sempre en l'ordre que marca la definició del mètode, definició que s'anomena *signatura* o *"prototipus"*, (*prototype)*. El prototipus de la funció de l'exemple serà:

```
public static int maxim (int a, int b);
```

A la crida de la funció, sempre s'hauran de passar tants valors (arguments) com paràmetres formals hi hagi a la definició de la funció cridada (prototipus). Aquests valors utilitats a la crida se n'aniran copiant sobre les variables definides com a paràmetres; ha d'existir una correspondència total tant entre el nombre de valors i el nombre de paràmetres, com entre els seus tipus. En el cas anterior, la crida des de el *main* serà:




```
public static void main( String[ ] args )
{
        int x, y, maxi = 0 ;
        Scanner teclat = new Scanner (System.in);
        System.out.println("Introdueix un numero: ");
        x = teclat.nextInt();
        System.out.println("Introdueix un altre numero: ");
        y = teclat.nextInt();
        maxi = maxim(x, y);
        System.out.println("El màxim es: "+ maxi);
}

public static int maxim ( int a, int b )
{
        int max;

        if (a<b) max=b;
        else max=a;
        return (max);
}
```

La definició de la funció maxim haurà de trobar-se dins de la classe on tenim el main, malgrat més endavant veurem que això pot canviar.

- ***Mètodes sense retorn: Accions o procediments***

Les accions es poden considerar com un cas particular de les funcions, són funcions en les que no hi ha valor retornat i per tant no hi ha *return.* En aquest cas les el tipus de dada retornat per la funció és *void* i quedarà:

<tipus_acces> <void> <nom_de_la_funció> (*llista_de_paràmetres*)

{

cos_de_la_funció ;

}

Per exemple:

```
public static void main(String [] args) {
        int x, y, maxi = 0;
        titol();
        Scanner teclat = new Scanner (System.in);
        System.out.println("Introdueix un numero: ");
        x = teclat.nextInt();
        System.out.println("Introdueix un altre numero: ");
        y = teclat.nextInt();
        maxi = maxim(x, y);
        System.out.println("El maxim es: " + maxi);
}
public static int maxim ( int a, int b ) {
        int max;
        if (a<b) max=b;
        else max=a;
        return max;
}
public static void titol( ) {
        System.out.println("PROGRAMA DE PROVES");
        System.out.println("Proporciona el màxim de dos numeros enters " );
}
```

Aquests tipus de mètodes seran sempre tasques concretes que no proporcionen un resultat, com per exemple mostrar alguna cosa per pantalla com potser un menú.

- ***Paràmetres d'un mètode***

Els mètodes en Java treballen amb dos tipus de dades:

- Variables locals: declarades en el cos del mètode. Només tenen validesa dins del mètode i es creen i es destrueixen amb l'execució del mètode.
- Paràmetres: Són els que permeten la comunicació del mètode amb la resta del programa mitjançant la transferència de dades. En Java es passen sempre per valor, es a dir a la crida del mètode es passa només el valor (una còpia o rèplica) de la variable.

En el exemple:

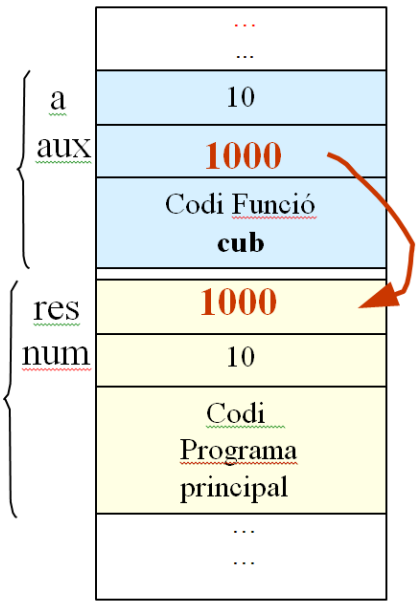
```
public static void main(String [ ] args) {
        int num, res;
        Scanner teclat = new Scanner (System.in);
        System.out.println("Introdueix un numero: ");
        num = teclat.nextInt();
        res = cub ( num );
        System.out.println("Resultat: " + res);
}

public static int cub (int a) {
        int aux;
        aux = a*a*a ;
        return aux;
}
```

Paràmetre per valor

Variable local

El programa principal crida al mètode *cub* i s'interromp per començar la execució d'aquest mètode, llavors es reserva memòria per a el codi del mètode, per a les variables locals (aux) i per als paràmetres (a):



MEMORIA

Qualsevol modificació en el valor dels paràmetres no es mantindrà quan acabi l'execució del mètode.

- ***Funcions principals de l'API de Java***

Les sigles API tenen el seu origen en *Application Programming Interface* i consisteix en un conjunt de llibreries de codi Java compilat o classes ofertes per la companyia Oracle i llistes per que siguin utilitzades per tots els desenvolupadors i/o programadors.
Aquestes "biblioteques de classes" són utilitzades en la gran majoria de llenguatges orientats a objectes, facilitant així el treball amb aquests llenguatges de programació. L'API de Java presenta una immensa quantitat i varietat de paquets i classes, cadascun destinats a una funció determinada.

Alguns són:

- "lang" : inclou classes bàsiques o essencials. És l'únic paquet que s'inclou de manera automàtica a qualsevol programa Java i consta de strings, números, threads i seguretat.
- "net" : proveeix suport per a sockets i conté classes com IP, TCP, URL o UDP entre d'altres que permeten la implementació d'aplicacions distribuïdes.
- "io" : inclou classes per a la serialització d'objectes i manejar l'entrada i la sortida.
- "awt" : conté classes que permeten pintar imatges i gràfics, manejar la GUI.
- "swing" : igual a l'awt però més moderna.
- "sql" : inclou classes que permeten el maneig de bases de dades relacionades.
- "util" : facilita i permet el maneig d'hores, dates, excepcions, estructures de dades o col·leccions a través de les classes que l'integren.

Hi ha moltíssims més API's de Java pel que us recomanem visitar la documentació aportada per Oracle (https://docs.oracle.com/javase/7/docs/api/)

- ***Consideracions sobre les funcions (mètodes)***

  - ✓ Les funcions han de ser el més independent possible de la resta del programa, i per això que només es poden comunicar amb la resta del programa mitjançant els paràmetres. **NO es poden utilitzar variables globals**.
  - ✓ Una funció es crida sense un objecte creat (ex: Math.pow()), un mètode sempre precisa d'un objecte prèviament creat (ex: alumnes.imprimir())
  - ✓ A la documentació del programa, se haurà d'escriure per a cada funció, el seu nom, els paràmetres d'entrada i sortida que en té i una breu descripció del què fa. Es pot fer servir els comentaris de la pròpia funció.

- ***Recursivitat***

Parlem de recursivitat quan un funció es crida a si mateixa.

La recursivitat pot substituir la iteració en el disseny d'un programa, però s'ha de tenir en compte que no sempre és la solució més eficient, ja que les crides recursives precisen de molts recursos, malgrat en molts casos si que representa la solució més senzilla per implementar.

En el disseny de programes recursius hem de:

1.- Identificar el cas bàsic o de sortida

2.- Determinar la manera de resoldre el cas general

3.- Comprovar que els diferents casos recursius es vagin apropant al cas bàsic.

El cas típic és el càlcul del factorial, una solució iterativa serà:

```
public static int factorial(int n) {
        int f = 1;
        int i = 0;
        while(i < n) {
                i = i + 1;
                f = f*i;
        }
        return f;
}
```

Que podem transformar en solució recursiva tenim en compte que:

Factorial: $n!=n\cdot(n-1)\cdot(n-2)\cdots 2\cdot 1$

Definició recursiva: $n!=n\cdot(n-1)!$, $n > 0$ / $n = 0$, n! = 1

```
public static int factorial(int n) {
        if(n == 0) return 1;
        else return n*factorial(n -1);
}
```

BASE PRÀCTICA

- **Programes d'exemple**

1.- El següent programa mostra un menú d'una calculadora elemental (suma i resta) i l'usuari ha de triar l'opció.

a) Solució directa.

```
import java.util.Scanner;

public static void main(String[ ] args) {

    int opcio = 0;
    double op1,op2,result;

    Scanner teclat = new Scanner (System.in);

    System.out.println("MENU D'OPCIONS");
    System.out.println("1. Suma");
    System.out.println("2. Resta");
    System.out.println("3. Sortir");
    System.out.println("Seleccioneu una opcio: ");

    opcio = teclat.nextInt();
    System.out.println("L'opcio triada es: " + opcio);

    if(opcio == 1) {
            System.out.println("Introdueix el primer operand: ");
            op1 = teclat.nextDouble();
            System.out.println("Introdueix el segon operand: ");
            op2 = teclat.nextDouble();
            result = op1 + op2;
            System.out.println("El resultat de la suma es: "+ result);
    }else if(opcio == 2) {
            System.out.println("Introdueix el primer operand: ");
            op1 = teclat.nextDouble();
            System.out.println("Introdueix el segon operand: ");
            op2 = teclat.nextDouble();
            result = op1 - op2;
            System.out.println("El resultat de la resta es: "+ result);
    }else if(opcio == 3) {
            System.out.println("ADEU");
    }
}
```

b) Solució amb una acció:

```
    import java.util.Scanner;

    public static void main(String[ ] args) {

        int opcio = 0;
        double op1,op2,result;

        Scanner teclat = new Scanner (System.in);

        menu();

        System.out.println("Seleccioneu una opcio: ");
        opcio = teclat.nextInt();
        System.out.println("L'opcio triada es: " + opcio);

        if(opcio == 1) {
                System.out.println("Introdueix el primer operand: ");
                op1 = teclat.nextDouble();
                System.out.println("Introdueix el segon operand: ");
                op2 = teclat.nextDouble();
                result = op1 + op2;
                System.out.println("El resultat de la suma es: "+ result);
        }else if(opcio == 2) {
                System.out.println("Introdueix el primer operand: ");
                op1 = teclat.nextDouble();
                System.out.println("Introdueix el segon operand: ");
                op2 = teclat.nextDouble();
                result = op1 - op2;
                System.out.println("El resultat de la resta es: "+ result);
        }else if(opcio == 3) {
                System.out.println("ADEU");
        }
    }

public static void menu () {
    System.out.println("MENU D'OPCIONS");
    System.out.println("1. Suma");
    System.out.println("2. Resta");
    System.out.println("3. Sortir");
}
```

c) Solució amb una acció i dues funcions:

```
import java.util.Scanner;

public static void main(String[ ] args) {
	int opcio = 0;
	double op1,op2,result;
	Scanner teclat = new Scanner (System.in);
	menu();
	System.out.println("Seleccioneu una opcio: ");
	opcio = teclat.nextInt();
	System.out.println("L'opcio triada es: " + opcio);

	if(opcio == 1) {
	    System.out.println("Introdueix el primer operand: ");
	    op1 = teclat.nextDouble();
	    System.out.println("Introdueix el segon operand: ");
	    op2 = teclat.nextDouble();
	    result = suma(op1,op2);
	    System.out.println("El resultat de la suma es: "+ result);
	}else if(opcio == 2) {
	    System.out.println("Introdueix el primer operand: ");
	    op1 = teclat.nextDouble();
	    System.out.println("Introdueix el segon operand: ");
	    op2 = teclat.nextDouble();
	    result = resta(op1,op2);
	    System.out.println("El resultat de la resta es: "+ result);
	}else if(opcio == 3) {
	    System.out.println("ADEU");
	}
}

public static void menu () {
	System.out.println("MENU D'OPCIONS");
	System.out.println("1. Suma");
	System.out.println("2. Resta");
	System.out.println("3. Sortir");
}

public static double suma (double x, double y) {
	double res;
	res = x + y;
	return res;
}

public static double resta (double x, double y) {
	double res;
	res = x - y;
	return res;
}
```

d) Solució amb tres funcions:

```
import java.util.Scanner;

public static void main(String[ ] args) {
    int opcio = 0;
    double op1,op2,result;
    Scanner teclat = new Scanner (System.in);
    opcio = menu();
    System.out.println("L'opcio triada es: " + opcio);
    if(opcio == 1) {
            System.out.println("Introdueix el primer operand: ");
            op1 = teclat.nextDouble();
            System.out.println("Introdueix el segon operand: ");
            op2 = teclat.nextDouble();
            result = suma(op1,op2);
            System.out.println("El resultat de la suma es: "+ result);
    }else if(opcio == 2) {
            System.out.println("Introdueix el primer operand: ");
            op1 = teclat.nextDouble();
            System.out.println("Introdueix el segon operand: ");
            op2 = teclat.nextDouble();
            result = resta(op1,op2);
            System.out.println("El resultat de la resta es: "+ result);
    }else if(opcio == 3) {
            System.out.println("ADEU");
    }
}

public static int menu () {
    int opc = 0;
    Scanner input = new Scanner (System.in);
    System.out.println("MENU D'OPCIONS");
    System.out.println("1. Suma");
    System.out.println("2. Resta");
    System.out.println("3. Sortir");
    System.out.println("Seleccioneu una opcio: ");
    opc = input.nextInt();
    return opc;
}

public static double suma (double x, double y) {
    double res;
    res = x + y;
    return res;
}
public static double resta (double x, double y) {
    double res;
    res = x - y;
    return res;
}
```

2.- Implementeu un programa per calcular la distància entre dos punts.

```
import java.util.Scanner;

public static void main(String[ ] args) {
   double a, b;
   Scanner teclat = new Scanner (System.in);
   System.out.println("Introdueix el primer punt: ");
   a = teclat.nextDouble();
   System.out.println("Introdueix el segon punt: ");
   b = teclat.nextDouble();
   System.out.println("Distància entre: " +a+ " i "+b+" es "+ distancia(a,b));
}

/*Funció: distancia
* Us:  f = distancia(p,q);
*       double f,p,q
* Funció que torna la distància entre p y q.*/

public static double distancia(double x, double y) {
   double d = x - y;
   if (d<0) d = -d;
   return d;
}
```

3.- Implementeu un programa amb la funció: *int sum_digit(int n);*
que calculi la suma del dígits d'un número enter.

```
import java.util.Scanner;

public static void main(String[ ] args) {
   int num;

   System.out.println("Programa que calcula la suma dels seus digits ");
   System.out.println("Introduiu un numero enter positiu: ");

   Scanner teclat = new Scanner (System.in);
   num = teclat.nextInt();

   System.out.println("La suma dels dígits del numero: " + num + " es " +
                                    sum_digit(num) );
}

public static int sum_digit(int n) {
   int suma = 0;
   if (n < 0) n = -n;
   while (n!=0) {
          suma = suma + n%10;
          n = n/10;
   }
   return suma;
}
```

- **Programes per completar**

1.- Modifiqueu el programa 2 per calcular la distancia entre dos punts de l'espai a partir de les seves coordenades: (x_0, y_0) , (x_1, y_1)

2.- Modifiqueu el programa 3 de manera que es calculi el número resultant d'escriure en ordre invers les xifres d'un número enter. Per exemple*, invert*(321) = 123.

- **Programes per desenvolupar**

1.- Modifiqueu el programa 2 per calcular la distancia entre dos punts de l'espai a partir de les seves coordenades: (x_0, y_0) , (x_1, y_1) i creant una funció per calcular la potencia d'un número.

2.- Implementeu un programa amb una funció: *bool es_primer(int n)* que ens confirmi si un nombre es primer o no.

- **Programes opcionals**

1.- Escriu un programa que calculi quantes combinacions de n elements agrupats de m en m són possibles en base a la formula : $C_m^n = m! / n! (m-n)!$
Divideix el programa en funcions.

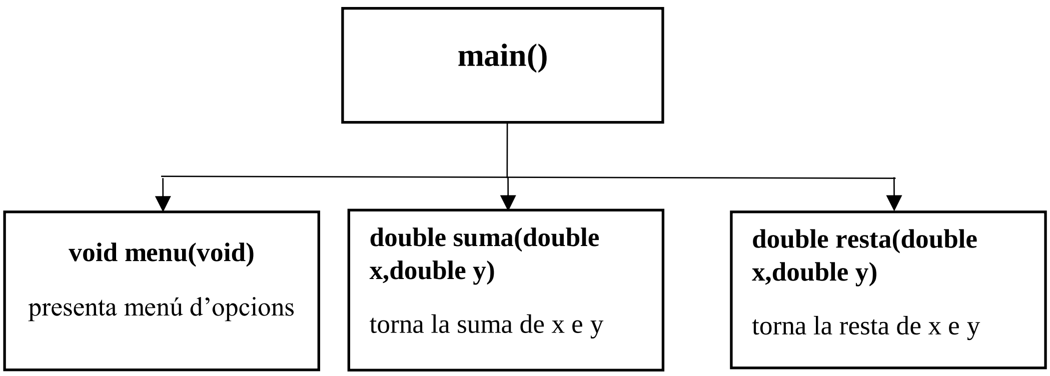
2.- Escriu un programa que mostri diferents felicitacions per pantalla. L'usuari ha de poder triar la felicitació des d'un menú amb diferents tipus de felicitació: d'aniversari, de Nadal, d'any nou, etc. Les felicitacions han de muntar-se a base de caràcters i símbols:

```
& & & & & & & & & & & & & & & & & & & & & & & & & & & & & & &
& * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * &
& * *                    FELIÇ ANIVERSARI!!!!                    * *&
& * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * &
& & & & & & & & & & & & & & & & & & & & & & & & & & & & & & &
```

S'han d'utilitzar una funció o acció per a cada felicitació.

- **Miniprojecte (OBLIGATORI)**

Dissenyar i implementar un programa que seguint el model del programa de l'exemple 1 presenti un menú amb les operacions d'una calculadora científica. S'ha de fer un disseny modular coherent amb l'estructura i funcionament del programa. S'ha de presentar també el diagrama "top-down" del disseny presentat.



- **Programes del jutge**

De l'apartat *"Primers procediments"* heu de fer entrega d'un mínim de 8 exercicis.

Els recomanats són:

P96275 Absolute value

P57846 Maximum of two integer numbers

P73231 Maximum of four integer numbers

P57474 Iterative factorial

P17913 Iterative double factorial

P55722 Iterative number of dígits

P77686 Palindromic numbers

P95401 Function for leap year